AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method of creating run time executable code for a processing element array, comprising:

partitioning the processing element array into a plurality of hardware accelerators;

identifying a plurality of functions in a program source code ~~that are anticipated to consume a substantial execution time~~ based on one or more run-time attributes of the plurality of functions;

~~decomposing~~ separating the identified plurality of functions from the program source code ~~into~~ in a plurality of kernel sections~~, wherein the identified plurality of functions are recognized as the plurality of kernel sections~~;

producing the run time executable code including a plurality of hardware dependent executable code portions that correspond, respectively, with the identified plurality of functions;

mapping said plurality ~~of kernel sections into a plurality~~ of hardware dependent executable code portions to ~~for execution on~~ the plurality of hardware accelerators; ~~and~~

identifying a plurality of hardware variants in the mapped plurality of hardware dependent executable code portions, wherein at least two of the hardware variants have different hardware configurations that are substantially functionally equivalent; and

~~forming~~ producing a matrix ~~describing~~ that defines different combinations of said plurality of hardware accelerators, ~~code~~ said hardware variants and said hardware dependent executable code portions, the matrix being configured to reference the mapped plurality of hardware executable code portions during run time execution of the plurality of hardware executable code portions. ~~to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.~~

2. (Original) The method of claim 1, wherein said partitioning includes partitioning into digital signal processors.

3. (Original) The method of claim 1, wherein said partitioning includes partitioning into bins.

4. (Previously Presented) The method of claim 1, wherein said mapping includes mapping into multiple hardware contexts.

5. (Original) The method of claim 4, wherein said mapping into multiple hardware contexts includes mapping a first set of variants.

6. (Original) The method of claim 5, wherein said first set of variants are produced based upon resource usage.

7. (Previously Presented) The method of claim 5, wherein said mapping includes mapping a second set of variants configured to support multiple hardware configurations of one of a plurality of bins.

8. (Original) The method of claim 1, wherein said mapping is performed by a place and route.

9. (Currently Amended) The method of claim 1, wherein said ~~decomposing~~ separating is performed manually.

10. (Currently Amended) The method of claim 1, wherein said ~~decomposing~~ separating is performed by a software profiler.

11. (Currently Amended) The method of claim 10, wherein said ~~decomposing~~ separating includes executing code compiled from said program source code and monitoring timing of said executing.

12. (Original) The method of claim 11, wherein said executing utilizes a set of test data.

13. (Original) The method of claim 11, wherein said monitoring includes determining functions that consume a significant portion of said timing of said executing.

14. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions by identifying regular structures.

15. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions with a limited number of inputs and outputs.

16. (Previously Presented) The method of claim 1, wherein said identifying includes identifying functions with a limited number of branches.

17. (Currently Amended) The method of claim 1, ~~wherein decomposing identifies~~ further comprising identifying overhead sections of the source code.

18. (Original) The method of claim 1, wherein mapping includes creating microcode.

19. (Original) The method of claim 1, wherein said mapping includes creating context dependent configurations.

20. (Original) The method of claim 1, wherein said matrix is sparsely-populated.

21. (Original) The method of claim 1, wherein said matrix is fully populated

22. (Currently Amended) A system for creating run time executable code for execution on a processing element array, comprising:

a plurality of hardware accelerators ~~partitioned from~~ defined in the processing element array;

a plurality of kernel sections of a program source code, the kernel sections including functions identified based on one or more run-time attributes of the functions; ~~as functions that are anticipated to consume a substantial execution time of a program source code when executed on said plurality of hardware accelerators;~~

a plurality of hardware dependent executable code portions derived, respectively, from said kernel sections, the hardware executable code portions being adapted for mapping to and execution on said plurality of hardware accelerators; ~~and~~

a plurality of hardware variants included in the mapped plurality of hardware dependent executable code portions, wherein at least two of the hardware variants comprise different hardware configurations that are substantially functionally equivalent; and

a matrix ~~describing~~ defining different combinations of said hardware accelerators, ~~code~~ said hardware variants and said hardware dependent executable code portions, the matrix being configured to reference the mapped plurality of hardware executable code portions during run time execution of the plurality of hardware executable code portions. ~~configured to support run time execution on the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.~~

23. (Previously Presented) The system of claim 22, wherein said hardware accelerators include digital signal processors.

24. (Previously Presented) The system of claim 22, wherein said hardware accelerators include bins.

25. (Previously Presented) The system of claim 24, wherein said bins support multiple hardware contexts.

26. (Original) The system of claim 25, wherein said bins support a first set of variants configured to support said multiple hardware contexts.

27. (Original) The system of claim 26, wherein said first set of variants are produced based upon resource usage.

28. (Previously Presented) The system of claim 27, wherein a second set of variants is configured to support multiple hardware configurations of one of said bins.

29. (Original) The system of claim 22, wherein said mapping is performed by a place and route.

30. (Currently Amended) The system of claim 22, wherein said ~~decomposing is performed~~ kernel sections are derived manually.

31. (Currently Amended) The system of claim 22, wherein said ~~decomposing is performed by~~ kernel sections are derived using a software profiler.

32. (Previously Presented) The system of claim 31, wherein said software profiler executes code compiled from said program source code, and monitors time consumed.

33. (Original) The system of claim 32, wherein said software profiler includes a set of test data.

34. (Original) The system of claim 32, wherein said software profiler determines functions that consume a significant portion of said time consumed.

35. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying regular structures.

36. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying sections with a limited number of inputs and outputs.

37. (Original) The system of claim 31, wherein said software profiler is configured to identify kernel sections by identifying sections with a limited number of branches.

38. (Original) The system of claim 31, wherein said profiler identifies overhead sections.

39. (Previously Presented) The system of claim 22, wherein said hardware dependent executable code includes microcode.

40. (Original) The system of claim 39, wherein said microcode includes context dependent configurations.

41. (Original) The system of claim 22, wherein said matrix is sparsely-populated.

42. (Original) The system of claim 22, wherein said matrix is fully populated.

43. (Currently Amended) A machine-readable medium having stored thereon instructions for execution by a processing element array, which when executed by said processing element array perform the following:

partitioning the processing element array into a plurality of hardware accelerators;

identifying a plurality of functions in a program source code ~~that are anticipated to consume a substantial execution time~~ based on one or more run-time attributes of the plurality of functions;

~~decomposing~~ separating the identified plurality of functions from the program source code ~~into~~ in a plurality of kernel sections~~, wherein the identified plurality of functions are recognized as the plurality of kernel sections~~;

producing the run time executable code including a plurality of hardware dependent executable code portions that correspond, respectively, with the identified plurality of functions;

mapping said plurality ~~of kernel sections into a plurality~~ of hardware dependent executable code portions to ~~of execution on~~ the plurality of hardware accelerators;~~and~~

identifying a plurality of hardware variants in the mapped plurality of hardware dependent executable code portions, wherein at least two of the hardware variants have different hardware configurations that are substantially functionally equivalent; and

~~forming~~ producing a matrix ~~describing~~ that defines different combinations of said plurality of hardware accelerators, ~~code~~ said hardware variants and said hardware dependent executable code portions, the matrix being configured to reference the mapped plurality of hardware executable code portions during run time execution of the plurality of hardware executable code portions. ~~to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.~~

44. (Currently Amended) A system configured to create run time executable code for execution by a processing element array, comprising:

means for partitioning the processing element array into a plurality of hardware accelerators;

means for identifying a plurality of functions in a program source code ~~that are anticipated to consume a substantial execution time~~ based on one or more run-time attributes of the plurality of functions;

means for ~~decomposing~~ separating the identified plurality of functions from the program source code ~~into~~ in a plurality of kernel sections~~, wherein the identified plurality of functions are recognized as the plurality of kernel sections~~;

means for producing the run time executable code including a plurality of hardware dependent executable code portions that correspond, respectively, with the identified plurality of functions;

means for mapping said ~~plurality of kernel sections into a~~ plurality of hardware dependent executable code portions to ~~for execution on~~ the plurality of hardware accelerators; ~~and~~

means for identifying a plurality of hardware variants in the mapped plurality of hardware dependent executable code portions, wherein at least two of the hardware variants have different hardware configurations that are substantially functionally equivalent; and means for ~~forming~~ producing a matrix ~~describing~~ that defines different combinations of said plurality of hardware accelerators, ~~code~~ said hardware variants and said hardware dependent executable code portions, the matrix being configured to reference the mapped plurality of hardware executable code portions during run time execution of the plurality of hardware executable code portions. ~~to support run time execution of the plurality of kernel sections by the processing element array, wherein each code variant performs a function whose inputs and outputs are identical.~~